Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

AD-A239 517

(2)

DTIC ELECTE AUG 16 1991 S D

REPORT DOCUMENTATIO

pproved
o 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS None
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report No. 4	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION University of Pittsburgh	6b OFFICE SYMBOL (If applicable)
7a NAME OF MONITORING ORGANIZATION Cognitive Science Program Office of Naval Research (Code 442CS)	
6c ADDRESS (City, State, and ZIP Code) 3939 O'Hara Street Pittsburgh, PA 15260	7b ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217
8a NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-K-85-0664	
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
61153N	RR04206	RR04206-0C	702-13

11 TITLE (Include Security Classification)
A Tale of Two Settings: The Lab and the Classroom

12 PERSONAL AUTHOR(S)
Janet W. Schofield

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM 1/91 TO 8/91	1991 August 8	46

16 SUPPLEMENTARY NOTATION
None

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	classroom social behavior, teachers' roles, peer helping, motivation, computer programming, computer science classes
05	08		

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Computer science students were observed in the classrooms of five different teachers in an intensive two year qualitative study. Observations and interviews led to the conclusion that students reacted very differently to the time they spent in the computer lab working on writing programs and the time they spent in the classroom where they learned about computers and programming through teacher-led lectures. Specifically, they enjoyed the lab more and were much more highly motivated to work in that setting. Analysis of the social processes in these two contrasting settings suggests that students' increased motivation in the lab was the result of a complex set of factors including (a) a shift in their relation to the teacher, (b) a concomitant shift in their relationship with peers, and (c) a shift in their relationship to the work.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
Dr. Susan Chipman	202-696-4318	ONR 442CS

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

91-08018

A Tale of Two Settings: The Lab and the Classroom

Janet W. Schofield
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, PA 15260

August 1991

Technical Report No. 4

This research was sponsored by the Cognitive Science Program, Cognitive and Neural Sciences Division, Office of Naval Research, under Contract No. N00014-K-85-0664, Contract Authority Identification Number NR 702-013.

The past decade has seen an incredibly rapid proliferation of microcomputers in both elementary and secondary schools. For example, between 1981 and 1984 the number of schools with microcomputers more than tripled (Quality Education Data, 1984). By 1985 almost all secondary schools and five-sixths of all elementary schools in the U.S. had at least some computers for use in instruction (Becker, 1986) and the trend toward the continuing acquisition of computers has continued. Current estimates are that more than two billion dollars have been spent to provide schools in the U.S. with computer technology in a period when school systems are under heavy pressure to spend their limited resources on numerous other things, including increased salaries for teachers (Buckley, 1988).

Although the remarkable rapidity with which microcomputers are being placed in schools is obvious, the impact of this change on teachers and students is not. In fact, our knowledge of the way in which this change influences classroom structure and functioning is extremely limited (Sheingold, Kane, & Endreweit, 1983; Sheingold, Martin, & Endreweit, 1987) and some studies suggest that the impact of educational software can be quite different from that which its developers intended (Hativa, Swisa, & Lesgold, 1989). Thus we decided to conduct an intensive qualitative study examining a wide variety of computer usage in a single school. Such an approach allows exploration of the extent to which different kinds of computer usage have similar or different effects as well as analysis of what the impact of any particular use may be.

The core of this study is an examination of the impact of one unusual but potentially very important usage of microcomputers -- their utilization as intelligent tutors -- on classroom structure and functioning in an urban school with a diverse student body. This research

has been reported elsewhere (Schofield, Evans-Rhodes, & Huber, 1989; 1990). However, we also closely examined computer usage in almost all other sites in the school in which computers were used regularly for instruction. The purpose of this was two fold -- to learn about how these very different kinds of computer applications effected classroom processes and to shed light on the question of the extent to which these effects were similar to or different from those of the artificially intelligent tutor.

The part of the project reported here closely examines social processes in computer science classes. This report focuses on a phenomenon which is in some ways parallel to one occurring in the GPTutor classes -- in both cases there was a marked increase in student interest and motivation when students were working on their computers compared to when they were not. The goal of this report is to discuss why this change occurred in the computer science lab. Although some reference is made to the findings on the impact of the GPTutor, a full analysis of the similarities and differences of the two situations is not the goal of this report. Rather I shall lay the groundwork for such a comparison at a later point by presenting a fairly detailed look at the computer science classes themselves. Before proceeding to this, however, I will briefly discuss the research methods used in this study.

Research Methods

The major methods of data-gathering employed in this study were intensive and extensive classroom observation and repeated extended interviews with students and teachers. Classroom observers used the "full field note" method of data collection (Olson, 1976) which involves taking extensive handwritten notes during the events being observed. Shortly thereafter, these notes are dictated into a tape recorder and then

transcribed. Observers made the field notes as factual and as concretely descriptive as possible to help avoid unwarranted inferences.

One clear problem with the use of such notes as a data base is what Smith and Geoffrey (1968) have termed the "two-realities problem" -- the fact that the notes as recorded cannot possibly include literally everything that has transpired. Hence, a source of potential bias is the possibility of selective recording of certain types of events. Although this problem is impossible to surmount completely in qualitative observation, there are some steps that can be taken to minimize its negative effect. For example, we found it useful to have two researchers observe a single setting both simultaneously and at different times. Discussion of differences between the two observers' notes helped to point out individual biases and preconceptions. Another technique useful in reducing the effect of such biases is actively to seek out data that undercut one's developing assessment of a situation. These techniques plus a number of others discussed in standard texts on qualitative research in educational settings (Bogdan & Biklen, 1982; Goetz & LeCompte, 1984) were employed to reduce the "two realities problem." Fuller discussion of methodological details can be found in Schofield (1985).

As mentioned previously, virtually all kinds of classes in which computers were used in instruction were observed during both years of the study, resulting in a very large data base gathered during almost 500 hours of classroom observation. With the exception of the classes using the GPTutor and selected comparison and control classes, the computer science classes received more intensive observation than any other single type of class. Specifically, we made observations weekly for two years in computer science classrooms at Whitmore. Mr. Brice, who taught by far

the largest number of computer science classes was observed most heavily. However, observations, generally weekly, were also made in the classes of four other computer science teachers -- Ms. Brown, Mr. Colgate, Mr. Davidson, and Mr. Deck.

Observers, no matter how omnipresent or insightful, are at a great disadvantage if they do not test their emerging ideas through direct inquiry with those whom they are observing. Because interviews can be so useful in providing the participant's perspectives on events, both formal and informal interviews were the second major data-gathering technique utilized in the research. The computer science teachers were interviewed both formally and informally. A randomly selected group of computer science students (N = 36) were also interviewed. (This constituted over 85% of the students asked to be in the sample.) These 40 minute interviews were semi-structured, consisting almost exclusively of open-ended questions. In constructing and conducting all interviews strong efforts were made to procure valid and unbiased data. For example, questions were posed in a balanced manner so that leading questions were avoided, students were assured that their teachers would not have access to any of their responses, and the like. All formal interviews of both teachers and students were taped and transcribed.

Data Analysis

Briefly describing data analysis procedures in qualitative research is extremely difficult since the process is so complex and iterative. To summarize, observational notes were coded as described in sources like Miles and Huberman (1984) and Strauss (1987). This involves carefully reviewing field notes as they are collected, creating coding categories of various types, developing and refining coding systems, writing working memos, and then searching for ways to refute or enrich the ideas

developing from the preceding process. Interviews were analyzed using traditional content analysis procedures.

Three general principles guided both the data-gathering and the data analysis phases of the research. First, a concerted effort was made to be as rigorous and systematic as possible. For example, sampling techniques were employed where appropriate; trained coders coded open-ended interviews using reliable systems developed for this research; field notes were carefully indexed so that all notes relevant to a given topic could be examined, etc.

Second, we took very seriously the importance of triangulating the data (Webb, Campbell, Schwartz, & Sechrest, 1966). That is, great care was taken to gather many different types of information bearing on the same issue, to minimize the potential problems with each data source, and to be sensitive in analyzing and interpreting the data to biases which could not be completely eliminated.

The third general principle which we took very seriously was that data analysis should be an on-going and iterative process. As the field notes and other data accumulated, they were indexed, read, and reread. Informal working memos were written, and data relevant to ideas emerging from the early stages of analysis were actively sought in planned and systematic ways.

Findings

Students enrolled in computer science at Whitmore were generally enthusiastic about it. For example, when asked whether or not they were glad they were taking the course over 90% of the students interviewed replied affirmatively, and most were quite enthusiastic. Yet observation of these classes and more differentiated questions about reactions to the courses revealed that computer science courses were conducted in two very

different kinds of settings, the classroom and the computer lab, and that students' attitudes toward learning in these two different settings varied dramatically. This report briefly describes the way in which computer science courses were organized at Whitmore. Then, it documents the students' differential reaction to the two different class settings and explores why students' reactions to the two complementary parts of the course varied so radically. It concludes that the utilization of computers in the lab led to often inadvertent changes in the way class was conducted which were highly motivating to students.

The Classroom and the Laboratory

The facilities for teaching computer science at Whitmore consisted of a regular-sized classroom and a similar adjourning room in which the computers and printers were located. The classroom was equipped, as were most other classrooms at Whitmore, with a blackboard across the front wall, a teacher's desk in front of the blackboard, and rows of chairs made with writing arms attached for students. Most of the top half of the wall separating the class and the room housing the computers, which was frequently referred to as "the lab," was made of glass to allow visual contact from one room to the other. A door through this wall allowed easy movement from one room to the other. Although the lab had blackboards on the front wall as well as one side wall, there was no special desk for the teacher. The computers were arrayed on four parallel rows of tables, one against the wall separating the classroom and the lab, two back-to-back rows down the middle of the room, and one final row abutting the wall farthest from the classroom. One chair was placed near each computer and the computer tables were large enough so that students had space for books or papers. In all but the largest classes, there were more computers than students. Although there was a door directly connecting

this room to the hallway, it was rarely used since students exited through the classroom.

All of the five computer science teachers observed at Whitmore made use of both rooms, although they differed markedly, as will be discussed later, in the proportion of time spent in these two locations. Not surprisingly, rather different kinds of activities were carried out in the two settings. Even more importantly, the social context of learning varied dramatically in the two settings. As indicated earlier, the students had very different reactions to the two different milieus and the reasons for this are the main focus of this report.

The classroom was used for five major purposes. First, teachers lectured students about topics such as the history of computers, the past and current uses of computers, and the various kinds of programming languages and their uses. Second, class time was sometimes used simply for reading. Students were not allowed to take their computer science textbooks home with them. (I was told this policy had been adopted because the $35 books were too expensive to have one for each of the students in the three or four sections of computer science taught each year.) Thus, to the extent that a teacher wanted students to use the textbooks at all it was necessary to do so during school time. Third, the classroom was the location in which tests and quizzes were administered. These tests often focused on the material presented in the lectures and/or the book. Fourth, teachers provided students with specific information about the programming language they were studying, BASIC in Computer Science 1 and PASCAL in Computer Science 2. So, for example, the first year students learned about READ statements in BASIC. This kind of information was generally presented in relatively small chunks immediately before students received an assignment utilizing it in a

program. Fifth, students were sometimes instructed to write programs they would later try out on the computers in the laboratory. With the exception of work on constructing programs, these kinds of activities were virtually never conducted in the room housing the computers.

In the classroom, the teacher's approach was similar to that traditionally found in academic classes at Whitmore. Thus, lectures were the primary vehicle of instruction. In addition, the teachers often posed questions and then selected the students who were allowed or required to try to supply the answers. The approved way to learn in such situations was to attend closely to the teacher and be responsive to the teacher's directions about taking notes, supplying answers, and the like.

In contrast to the classroom, the lab was used for working <u>with</u> the computers rather than learning <u>about</u> them. Typically the students' task in the lab was to create, debug, improve, and elaborate programs. Students generally sat in front of their computers working on a programming project assigned by the teacher. Different teachers varied somewhat in the programming tasks assigned. However, in the very beginning of the year it was common for teachers to have students write programs to perform fairly simple mathematical calculations, such as converting Fahrenheit temperatures to centigrade or computing payrolls. Quite soon students progressed to writing somewhat more complicated programs incorporating a greater variety of commands. They frequently had time periods of a week or more to complete these tasks. It was common for teachers to require that these programs have certain characteristics which demonstrated competence with specific skills, but to leave the precise nature of the program up to each student. So, for example, the teacher might tell students to do a program incorporating loops but leave the function the program performed up to the students.

In the computer lab teachers rarely tried to instruct the class as a whole. Just as the teachers of students using the GPTutors circulated and dealt with student's problems individually (Schofield, Evans-Rhodes, & Huber, 1989; 1990), so too the computer science teachers went from student to student in the lab as the need arose. This occurred quite spontaneously since different students had problems with different parts of their programs. Thus it would have been not only unnecessary but also cumbersome and inefficient to try to insure that every student listened to the teacher's interactions with all other students.

Students' Preference For the Lab

Students' strong preference for working with the computers in the laboratory rather than learning about them was obvious in their everyday behavior. For example, students would frequently ask the teacher if they could go to the lab and made negative comments if told that the class would not be going there. Requests to stay in the classroom to work were extremely rare. The students' clear preference for the lab is illustrated in the following field notes:

> He (a substitute teacher named Mr. Wilborn) says, "Who wants to go to the lab?" Hands shoot up around the classroom. Almost all of the students have raised their hands. A couple of cries like "Yeah!" and "Let's go" emanate from the back of the room. The teacher says, "All right. Let's get started." The students literally surge out of their seats into the lab . . . . Within one minute they are all seated in the lab with the computers on.

This same preference was evident in the interviews. When asked whether they preferred to spend their time in the computer science lab or the classroom over 80% of the students interviewed in both computer science

1 and 2 classes stated a clear preference for the lab. Not a single student reported preferring the classroom. Comments like the following were common:

I: Do students act differently in the classroom and the lab?

Tim: They're bored in the classroom. They love it in the lab.

Consistent with their obvious preference for the lab, students often evidenced an enthusiasm for and interest in their work in the lab rarely apparent in the classroom. Thus, for example, it was common for students to put their heads down on their desks in the classroom to rest whereas such behavior was extremely rare in the lab. In interviews they frequently made spontaneous distinctions about their reactions to the two settings, as is evident below:

Interviewer: Can you tell me how computer science is different from your other classes?

Renata: There's no class I'd rather be in . . . I could stay all day, as long as we're in the lab.

Mr. Brice noted, interestingly, that students known as troublemakers by teachers in other classes were not problems in computer science, especially in the lab. "Their personalities change," he asserted.

Of course the fact that the students preferred the lab to the classroom is not in and of itself evidence that the lab is necessarily a better environment for learning. It is certainly possible to learn without enjoying the process and to enjoy oneself without learning. Thus my goal here is not to argue that computer science should be taught exclusively in a laboratory setting just because students enjoyed it more. Rather it is to examine why the laboratory was, relatively speaking, so attractive to students and what the implications of the differences in the settings were for students' involvement with and

attitude toward their work. I will argue that the students' preference for the lab was not just a result of a preference for programming over learning about computers or the other activities conducted in the classroom, although that is undoubtedly a part of the explanation. Rather, the two different milieus created a different context for learning. Specifically, the students' relationship with their teacher changed as did their relationships with their peers as they moved from one setting to another. In addition, the relation between the students and their work underwent a major shift. I will argue that most of these changes had a very positive effect on students' motivation. Lessons drawn from this analysis could well be applied to making classrooms a more stimulating and attractive environment -- a goal worthy of consideration in Whitmore and many other schools in which the drop-out rate is of serious concern and teachers often bemoan students' lackadaisical approach to their work.

The Basis for Students' Preference for the Lab

A changed relationship with the teacher. Students' relation to the teacher changed as they moved from the classroom to the lab. The shift in their relationship paralleled the change in teacher-student relationships when geometry students shifted from whole class instruction to work on the GPTutor (Schofield, Evans-Rhodes, & Huber, 1989; 1990). Specifically, in the computer lab the teachers functioned less as expert authority figures and more as skilled collaborators or coaches than they had previously. Students often remarked on this change. For example, when asked a very general question about how computer science classes were different from their other classes over two-thirds of the students in both computer science 1 and computer science 2 mentioned that their relation with their teacher was different, most commonly describing it

as less of an authority relationship or as more friendly. As one student put it, responding to a question about whether his relation with his computer science teacher was different from his relationship with other teachers.

> He don't treat us like we're students and he's the teacher . . . Most teachers think "I'm the teacher, you have to listen to me." Sometimes that irks people because they try to tell you how to do everything and some people don't like to be bossed around.

A large number of the students also characterized computer science teachers as more helpful than other teachers, echoing the observations of the student in the GPTutor classes quoted in an earlier technical report (Schofield, Evans-Rhodes, & Huber, 1989) who said of his teacher, "He doesn't teach us any more. He just helps us." As one computer science student put it:

> Usually he's helping people. Whereas most teachers stand up and talk at us, he comes around and actually sits down with you and tries to help you with your program, like individual help as much as he can. Most classes the teacher stands there and talks to you and you do your work and you hand it in and they give it back to you and that's it.

This shift occurred for many of the same reasons it did when the GPTutors were in use and the geometry students experienced a similar shift from lecture based whole class instruction to more individualized interactions with the teacher. As indicated earlier, a lecture format focuses attention on the teacher's superior knowledge since the lecture generally consists of information the teacher already knows that students are supposed to learn. Teachers exert a great deal of control over the topics

to be covered in lectures and, not surprisingly, tend to emphasize things they know a lot about and avoid areas in which their knowledge is less complete. Thus, in the classroom the teacher can generally provide a consistent display of knowledge superior to that of the students and sufficient to the task at hand. This was not possible to such an extent in the lab for reasons to be discussed shortly. In addition, since questions posed by the teacher during lectures generally concern specific facts, such as "What does the word binary mean?" or "What does the acronym ASCII stand for?", teachers were continually in the position of telling students whether they were right or wrong. Tests constituted yet another occasion in which the teacher's authority as arbiter of what is true was reinforced in the classroom.

Because the class constitutes an audience whose attention the teacher needs to direct and retain in order to achieve his or her goals during a lecture, the threat of distraction or disruption is a serious one. Thus rules against speaking without permission or moving out of one's seat are promulgated. Even the posing of too many unprompted questions by students may be discouraged if it interrupts the flow of the material planned by the teacher. Formal mechanisms for signaling a student's desire to speak, such as the raising of one's hand, allow the teacher to accommodate students' queries and comments without disrupting the on-going sequence of teacher-led activity. All in all, whole class instruction through lecturing creates a situation in which the teacher needs to maintain quite strict control over students' behavior. Attempts to achieve this control often involve threats of disciplinary action or grade reduction. Thus the authority that the teacher has solely by virtue of his or her position as teacher is often made quite salient.

The situation in the lab was quite different. There, the students' attention was not typically directed toward a teacher standing at the front of the room supplying them with information they were expected to learn. Rather, the student's task was to create working programs using whatever resources were available, including their own, their peers', and their teacher's knowledge, and programming manuals located in the room. Students worked on their programs as individuals, requesting the teacher's assistance when they felt they needed it.

In the lab teacher-student interactions were less likely to be authority-initiated demands for attention or information than in the classroom and more likely to be student-initiated requests for assistance. There it was relatively uncommon for the teachers to approach students and offer unsolicited advice. Rather, the need for assistance was frequently so great relative to the time the teacher had available that the teachers were kept busy responding to student initiated requests for help. When teachers had a temporary respite they tended to take care of paper work, work on their own programs, or even play computer games rather than to circulate offering unsolicited help. Computer science 2 classes were much smaller than the introductory course, generally consisting of fewer than a dozen students rather than one and a half or two dozen students. However, the teacher in charge of them, Mr. Brice, acted much as he and the other teachers did in computer science 1, partly because he felt that advanced students should be encouraged to work independently and partly because he enjoyed improving his programming skills by working on difficult problems.

In the lab even very knowledgeable teachers were often presented with problems which they had to think about. In fact, teachers could commonly be observed consulting programming manuals for information or

struggling to figure out how to solve a problem, behaviors which were much less common in the classroom. Teachers could decide what programs to have students work on in the lab in a way which reflected their own areas of expertise, just as they were free in the class to emphasize the topics with which they were most familiar. For example, teachers who had more experience with graphics than othe assigned more graphics programs. However, many students were sufficiently interested in the work that they went beyond the assignment's minimum requirements in ways which were not always easily predicted. In addition, there were a few students, generally white boys, for whom computers were a hobby of great personal interest. These students entered the computer science courses with a great deal of programming experience and were highly motivated to create rather elaborate programs, well beyond what the course required. Thus, it was not uncommon for students to ask questions or present programming problems which constituted a real challenge to their teachers. All this combined to create a rather different image for the teacher in the lab than in the classroom. Rather than being seen as a repository of an endless store of facts the teacher was seen as a sometimes fallable individual trying to apply and extend his or her knowledge, much as the students were.

In the lab students also had a very different way of judging the quality of their work which was much less dependent on the teacher's authority than it was typically in the classroom. Specifically, they could try their programs out and see if they functioned as they were intended to. The fact that they did so was independent confirmation that the programs worked. Their failure to do so was an objective indicator of the program's deficiencies. This tended to undercut the authority of the teacher as the final arbiter of "right" and "wrong" and "good" and

"bad." Students could see for themselves if their program functioned and whether the results were impressive or not. Thus, students received clear and immediate feedback without having to depend on the teacher's judgment. Interestingly, one negative consequence of this shift in the standard students used for judging their work was they were often not too interested in their teacher's advice on how to make the structure of a program more elegant or efficient. Idiosyncratic ways of doing things, which might be dysfunctional in the long run to the student's programming skill, were accepted as unproblematic by students, even though their teachers might try to point out ways which were better.

> Mr. Brice: I have a kid named Jim Chiu, whose father is at the university . . . Jim has his own brand of looping, which is very unique to him. When he helps other students it doesn't mesh into their programs very well. So, I kind of discourage other kids from accepting, point blank, his solutions to a problem . . . Occasionally I have to say, "Jim, don't help these people anymore, because you're not teaching them the way I would like them to be taught . . ." If they were in computer science 2 I would be able to explain it to them, but right now they think, "If it works, it works." They don't see why it would be nice if it had a nice running pattern to it. As long as it works, they figure it's good.

In fact, the acid test of whether or nct a program would work was applied not only to students' efforts but also to the teachers'. Thus, the teachers' skill in solving the problems students brought to them was constantly on trial. If a program would not run after a student followed a teacher's advice it was clear that the advice was deficient in some regard.

In cases where the teacher was knowledgeable, a very clear sense of colleagueship arose between students and the teacher. This kind of relationship was perhaps most evident between Mr. Brice and his students and appeared to be felt by teacher and student alike. Speaking about one of the computer hobbyists mentioned about, Mr. Brice said:

> I have a student . . . who is very talented in programming. If I had any problems or challenges I could give them to him and he could work them out. Between the two of us one would come up with a solution for it.

Students' remarks about Mr. Brice also reflect this sense of colleagueship and mutual exploration of their subject.

> I: Compared to other classes, how important is the teacher in helping you learn in computer science?
>
> Sarah: . . . When they share with you what they are learning it's important. He shares what we're learning because he learns from us too. We depend on him but he learns from us. Other teachers know everything. You can't argue with them. About dates, for example, they know! Mr. Brice is like "I don't know. I'll look in the book." Then if you're wrong, you're wrong. He listens to what you have to say.

The contrast between the teacher's role as a distant repository of authoritative information in the classroom and as a coach or skilled collaborator in the lab was apparent in all of the computer science classes observed. However, there were clear variations on this theme which depended heavily on the teacher's level of skill as a pedagogue, disciplinarian, and programmer. For example, although Mr. Davidson knew quite a bit about many aspects of programming, he had great difficult controlling the students, especially in the classroom. He greatly

preferred teaching evening classes at a local community college and conducting a consulting business -- both settings in which his demands for quiet attentiveness were more likely to be met. The high levels of tension and disrespect apparent in his classrooms were not conducive to the development of an easy colleagueship in the lab. Mr. Davidson often offered individualized assistance to students in the lab. However, he was sometimes prone to withdraw from them, even sitting out in the classroom catching up on paper work, including the grading of tests. In fact, on several days during which his class was observed working in the lab, Mr. Davidson spoke to no more than two or three students during the entire period, except for making an announcement at the beginning that students should go to the lab to work on programs.

In contrast, a number of other teachers' computer skills were too weak for them to consistently provide useful guidance when students faced difficult programming problems in the lab. This was hardly surprising since these teachers had been pressed into service in spite of the fact that they had little background in computer science because teachers were needed. A declining student population and a tight budget meant that rather than hiring new teachers with strong computer skills, Whitmore tended to use faculty from the math or science departments to teach computer science. Most of these individuals did not meet the district standards for certification as computer science teachers and their skills were often not adequate to the challenge of solving unexpected programming problems. One of these teachers said emphatically during an interview, "I hate computers and I hate not being able to help the students." Students were well aware of this lack of programming skill, using words like "quack" to describe such teachers.

One strategy for handling the dilemma of teaching something they really did not know very well was for teachers to spend a higher proportion of their time in the classroom, where, as has already been discussed, their lack of a strong working knowledge of programming was not such a handicap. Others avoided the lab in more unconventional ways, going on occasion as far as showing slides of a summer trip to far flung parts of the United States. Although students' lack of respect for such teachers was evident in interviews, they were generally reasonably compliant in the classroom which allowed the teachers to play out a fairly traditional role there. Typically such teachers adopted a more collegial manner in the lab. However, they were not dependably able to solve relatively straightforward problems which the better students could often handle easily.

> Linda shows Mr. Erie the listing of her program and Mr. Erie says, "Well it looks good to me; I don't know why it isn't working." He walks away. Mark (an average student) leans over to look at Linda's program. He points to one thing that is evidently wrong. Linda makes a change and then runs the program . . . She says to Mr. Erie, "Hey, it works now."

Students reacted negatively to the teachers being frequently unable to solve routine problems, thus suggesting that although they valued the sense of learning together with the teacher they experienced in the lab they, quite justifiably, desired a teacher who could act as a knowledgable guide or skilled colleague in the joint search for solutions rather than as a relatively uninformed peer.

A Changed Relationship With Peers

One factor which undoubtedly contributed to the shift in the relationship between students and teachers as they moved from the

classroom to the lab is that teachers did not enforce as many rules restricting students' freedom in the lab as they did in the classroom. In the classroom setting, where the student are an audience, it is distracting, even disruptive, for students to leave their seats or talk among themselves. Such behaviors make it hard for others to see and hear as they need to in order to follow the teacher's lesson. Teachers recognize this and use their authority to prevent or at least minimize these behaviors. In sharp contrast, the teachers' work in the lab, providing assistance to individuals who need it, is not likely to be hampered by other students leaving their seats or talking. Thus all of the computer science teachers appeared more tolerant of such behaviors in the lab setting, mitigating the distinction between teachers, who can move and speak as they please, and students who cannot.

The easing of such restrictions was clearly noted by students, who enjoyed the comparative freedom of the lab. Many took advantage of it to do a considerable amount of socializing. For boys, this often meant discussing sports. For girls, this was more likely to entail talking about other students, both male and female, and family members. However, students also took advantage of their freedom to move about and talk with others to obtain help with their work. In fact, when asked whether students helping behavior differed in computer science compared to other classes over three-quarters of the students said that students in computer science helped their peers more than in other classes. Students found this a very positive feature of working in the computer science lab, often making enthusiastic comments about it like the one below:

Interviewer: What is the best thing about taking computer science?

Carol: The students help each other. It's like teamwork. In other classes we don't get to do that . . . People have fun. They help each other out and I think that's great!

Many of the students explicitly linked the comparatively high rate of helping to the unusual degree of freedom in the lab to talk or move about with comments like:

Rich: They help each other more because they have the freedom to talk . . .

Interviewer: More than in other classes like English or geometry?

Ned: Yeah, because in most of the other classes they frown on people talking to each other. The teachers want total silence.

Of course, the fact that students were free to move and talk in a way that made it possible for them to get help from other students without resorting to subterfuge and breaking class rules does not mean that they would necessarily do so. In fact, a small number of students chose to work in an almost completely solitary fashion. Yet the large majority did seek and receive help from their peers frequently and seemed to feel quite positively about it. A number of factors seemed to be conducive to this development. First, it was a clear fact of life in the computer science lab that the teacher was often busy with other students. Thus, when the need for help arose students often had to wait quite a while if they insisted on making the teacher their only source of advice. Seeking help from a peer was often a very efficient way to proceed.

Ron: There should be more than one Mr. Brice. There should be two or three of him. He's always running around to a different person helping him out. He can't always get to a person (who

> needs help) . . . But if the person beside you knows what he's doing, it's all right.

Second, most students were sufficiently interested in their programs to want to get help when they were stuck rather than using the teacher's inability to help them immediately as an excuse for doing nothing or socializing for long periods of time. The words of one student clearly convey this widely shared feeling.

> You really have ambition to work in there (computer science). In other classes you just do what you have to do, but in here you want to make everything better. You don't just want to pass. You want to get an A+ on everything.

In most classrooms one major disadvantage of turning to peers rather than the teacher for help is that it is often hard to know how much credence to give their advice. For example, a peer can misspell a word, give one the incorrect formula for the area of a circle, or give bad advice on the organization of an essay. The student who needs help is often not in a good position to evaluate the quality of the advice received. However, in computer science students used a quick and efficient mechanism to evaluate advice on programming -- to try it and see how it worked.

There were two common patterns of peer help, reciprocal help between friends and help given by unusually knowledgeable students to a wide variety of others. First, it was common for friends or acquaintances to help each other, often as part of a reciprocal relationship in which help was sometimes given and sometimes received. One boy answered a question about which students work together in the following way.

> People who have stuff in common work together. Me, Dick, Bill, and Don are all athletes. We're all interested in football and

> baseball and we're always talking about everything. We just work together. And John too. Renata I've never worked with. Tonya, only a little bit.

Since patterns of social interaction within the school were heavily influenced by race and gender, such exchanges of assistance typically, though not always, occurred between students of the same sex and/or race. Often the giving and receiving of help was embedded in an on-going interaction which rapidly switched back and forth between causal socializing and a more task-oriented focus.

> Bill, who is white, is working on a game program. He asks Mark, who is also white, to help him finish it. They are joined by Doug and Martin, both of whom are black. Part of the time the boys collaborate on the program, often yelling loudly about whose statements are right. (In general this is good-natured with the students kidding each other about the particular way they go about solving programming problems.) They also discuss the dance that is scheduled for tonight and football.

A number of classes contained students known as "wizards" who were widely recognized as being unusually talented or experienced in computer programming. Such students, invariably male and usually white, often provided a great deal of assistance to students who knew less than they did. In many cases there seemed to be a tacit exchange of social acceptance for information received. As one student put it talking about a slightly built white male wizard:

> First when we found out Ned was good, people was kind of jealous . . . and talked about him. They got real upset just because he knows what he's talking about. Now it's okay since

> he's helping everybody. They thought he was going to be selfish about it.

Another white male who talked in an interview about the consequences of the fact that he and his friend were considered wizards said:

> They ask us how to do this or what you do in this case . . . They ask a lot of questions . . . Some students are really upset because we're in that class . . . But when they need help they're all real nice and friendly.

Sometimes the wizard's special competence allowed an ego-gratifying display of superiority. Such displays were not appreciated, but they were generally tolerated as the price one had to pay for expertise.

> Bill (a white "wizard") says to Don who is black, "You got a problem" when Don's chemistry programs says that water is a poison. Don lists his program and looks at it intently for a few minutes. Then he turns to Bill and says, "What's the matter here?" Bill says, "Okay. Let's see" . . . Bill starts troubleshooting, listing out the program, typing in changes, and the like. At one point Don tries to type something in on his keyboard and Bill says in an irritated tone, "Hold on a minute. Hold on a minute!" He then continues to study the program . . . He points to one section of it and says in a voice loud enough for the whole class to hear, "This is why. You don't have the locate statement like you should." Don says a bit sarcastically, "Well, sorry!" Bill replies in a cool tone, "You have to be intellectual about it."

The "wizards" were consistently able to bolster their self-esteem by helping others and generally did not resort to rubbing in their superior capability. The sense of accomplishment inherent in solving the problem

and the admiration of their prowess by the other students normally seemed to suffice. Of note is the fact that even less capable students often had the gratifying experience of solving a problem that stumped a peer. No doubt the frequency of this experience was raised by the fact that students often helped friends who tended to be at a roughly similar academic level. In addition, student's programs often failed to run because some minor convention, which could be spotted even by a relatively unskilled programmer, had been violated. As one student put it:

> Even the kid in the class who doesn't know anything knows something others don't know. It has happened to me (and) I'm the dumbest (in our class).

Although helping between students was a widespread phenomenon in the lab, generally accepted by students and teachers alike, there were some norms which regulated it. Teachers' attitude toward specific instances of such helping varied, depending to a large extent on whether they felt assistance was really needed. Furthermore, teachers disapproved of cases in which more advanced students literally took over and wrote major sections of programs for other students, rather than providing assistance with specific problems. Students saw nothing wrong with getting help from other students, including copying part of a program, when they were stuck. But they objected to "biting," copying part of someone's program without asking permission and/or acknowledging the assistance. They also objected to students who didn't honor the unwritten rules about reciprocity which required those who received help to return it if they could.

> Roberta: Dick . . . doesn't like sharing with Charlie because Charlie . . . just takes the examples . . . He doesn't give the input. He just takes the output.

Occasionally friction arose when the teacher or a student felt such norms were being violated. However, in general, peer helping interactions were positive in tone and contributed substantially to students' enjoyment of the lab.

These helping interactions also contributed to learning in at least three important ways. First, a request for assistance from a peer, often from a friend, provided substantial motivation to try to solve a problem. Students seemed to want to avoid letting their peers down in such situations. Thus, they generally worked fairly hard at solving problems brought to their attention. This provided a good opportunity for practicing their debugging skills and gaining new knowledge through trying out ideas, consulting a manual, or the like. Second, peers seemed to feel very free to discuss and evaluate each others' suggestions as they worked on solving problems. Sometimes this generated heated discussions, as indicated in the excerpt from the field notes appearing on page 18. The process of formulating and defending their ideas seems likely to help solidify a student's knowledge and clarify mistaken beliefs. Furthermore, even though their relationship with the teacher in the lab was collegial relative to the classroom, some students were inclined to accept a teacher's advice as likely to be right until proven otherwise. Peers' advice was often subjected to a more serious scrutiny, which called for more thinking and consideration of alternatives since the students felt freer to reject it. One student captured this sense of freedom when discussing peer helping by saying:

> They can help when the teacher is trying to get around. . . You can ask questions and they'll tell you. If you don't like it you can do it another way.

When rushed or too interested in their own programs or social conversations to want to converse at length, students did sometimes just allow others to copy from their programs without discussion or explanation. However, most students seemed aware that merely letting a friend copy was not conducive to learning and thus in the long run was not doing their friend a favor. At least some could articulate a conscious strategy of trying to get their peers to think. One student who helped others a great deal explained his approach to helping with math programs this way:

> When I help another student, I don't give him the answer. I'll write down the formula and ask him what he wanted to do . . . and then I'll tell him to try and figure out what is in the formula. If they get it wrong, I'll tell them what they have to figure out . . .

Finally, some students asserted that they preferred help from peers rather than from teachers since they could understand it better coming from someone with a level of knowledge or manner of speaking nearer their own.

In sum, students' freedom to interact more with their peers in the lab was a byproduct of the fact that the teachers did not lecture there. The individualized mode of instruction common in the lab made it unnecessary to forbid or strictly control student interaction. Indeed, the fact that the students' need for assistance could not be met promptly by one teacher encouraged teachers to allow students to help each other. Students used this freedom both to socialize and to help each other. The

socializing added an element of fun to their time in the lab which was not readily available in many other classroom settings. Although many students spent a substantial amount of time which could have been devoted to their lab socializing, most also showed a level of interest in their work in the lab which was not so readily apparent in the classroom. The giving and receiving of help often appeared to be quite effective both in encouraging students to think about what they were doing and in creating a positive attitude about the class.

> Interviewer: What's the best thing about computer science?
>
> Donna: That it's not so strict. You learn a lot more in a social environment with other students helping you out instead of just the teacher. I've learned a lot from other students. . . That's the best part.

A Changed Relationship Between Students and Their Work

Lab work more connected to students' career goals. Computer science is an elective course at Whitmore. Although some students reported enrolling in it primarily because it fit into their schedules or because of intrinsic interest in the subject matter, the most common reason given for enrolling in computer science was the belief that it would be of direct use in later education or in students' careers. A large number of students planned careers in fields as different as secretarial work, computer programming, and architecture in which the need for various kinds of computer skills is obvious. Many others, who were undecided about specific careers, were nonetheless confident that a knowledge of computers would be helpful in almost any field, as illustrated in the following excerpt from a student interview:

> Interviewer: Why did you decide to take Computer Science 1?

> Charlie: Well, for a lot of reasons. Computers is a growing industry and I figure if I take it now . . . I'll have a head start on whatever I want to do later.

Students' preference for working in the lab was linked to the fact that many of them believed that gaining experience with computers by learning programming would ultimately be more useful to them than most of the things they did in the classroom. However, this explanation is far from complete. Students needed to learn the kind of information about BASIC and PASCAL commands conveyed in the lectures if they were going to be able to program. They clearly recognized this, as evidenced by the fact that over 80% believed that these lectures were helpful to them in their later lab work. As one student put it in an interview:

> Students get mad when they have to go into the classroom because they want to get on the computers . . . But when Mr. Brice gives an assignment they're happy they was over there because they know what to do when they get on the computers.

Yet students were often very inattentive and restive during such lectures. In some classrooms, this restlessness sometimes progressed to open insolence when students were supposed to be reading or engaging in other particularly unpopular activities.

> The class is extremely rowdy now, with relatively few students reading . . . I (the observer) get hit by a flying spitball which was apparently intended for Ernie who is sitting near me. Mr. Davidson asks the students to be quiet saying, "There's only five minutes left. Let's get some work done." He then says to Ernie who is talking quietly, but audibly, "Did I ask you to be quiet?" Ernie says, "Sorry." Five or six "sorry's" echo through the room. Such echoing happens frequently in this

> class. For example, a few minutes ago Mr. Davidson explained something to a student and then said, "Do you understand? Are you following?" and around the room I heard four or five echoes of the same phrase . . . A male voice calls out loudly from the far side of the room, "Close those legs!" and giggles and laughs circulate around the room. Ernie is beet red in the face in what appears to be an effort to control the volume of his giggles. Mr. Davidson says repressively, "There's been a lot of unnecessary talking today."

Such behavior was much less common in the lab.

Lab work more connected to students' interests. In the classroom students were presented with facts to learn, much as was true in many of their other academic classes. Although many of these facts, such as information about the hexadecimal system, were basic to understanding how computers actually work, students tended to find them relatively uninteresting in comparison to programming in the lab as is clear from the following excerpt from a student interview.

> The teacher said at the beginning, "You're going to learn about the computer." I was like, "I don't want to learn about the computer. I just want to use it!"

One major attraction of lab work, compared to classroom work, was the degree to which students could link the work to their own personal interests and fantasies. As one student put it:

> You gotta do what he says. You gotta do the program he wants you to do, but . . . you can write things that are your creativity . . . You can put in parts that are from you. In other classes you don't have that freedom.

This freedom to "put in parts that are from you" was highly motivating to students. For some students, generally boys, this meant creating programs that kept track of information on sports teams or raced cars across the screen. Reflecting traditional sex roles to a striking degree, girls were much more likely to create programs that dealt with personal relationships. For example, a number of girls seemed fascinated by endless variations on a program constructed to flash their own and their boyfriends names on the screen like the one described below.

> Marta: I made this cute little thing and it asked for your name and your boyfriend's name. Then it prints little hearts. I did that . . . I did that and I'm so happy about it!

The kind of material covered in the classroom was less able to be melded readily with personal non-academic concerns and fantasies. Thus, it was far less appealing to most students. In addition, students were motivated by the sense of freedom and control the ability to link work in the lab to their interests gave them, relatively to many other school settings, including the computer science classroom, where they experienced much less of a sense of control over their environment. This feeling of control and the resultant personalization or ownership of the product is apparent in the following excerpt from an interview with a student.

> Interviewer: Are you glad you are taking computer science 1 or not?
>
> Sara: I'm glad. It's fun. You can make your computer do what you want to do. You can put what you want to put on it. You have your own disk and your own computer. It's fun.

Lab Work Requires Active Experimentation Rather Than Passive Assimilation

When asked how being in the lab differed from being in the classroom over 80% of the students spontaneously mentioned the contrast between the passive assimilation of knowledge characteristic of the classroom and the active involvement in learning typical of the lab. They overwhelmingly preferred a sense of active involvement in learning. As one student put it pithily, explaining why he liked the lab better, "You ain't got to listen to the teacher talk." Other students complained that taking notes on the teacher's lecture or reading the text was just plain boring. In contrast, working on the computer to develop and debug programs was generally seen as much more enjoyable and exciting. Comments like the following about computer science, and most especially about the lab were common.

> Interviewer: How would you rate what you learned in computer science compared to what you learned in other classes?
>
> Eric: I think I learned more in computer science 'cause it was different. Instead of just reading something out of a book and remembering it -- that's what you normally do in other classes -- in computer science you just work on it until you know it.

Another student expressed a similar sentiment like this:

> Interviewer: How is computer science different from your other classes?
>
> Elsie: A lot of people are used to the chalkboard method where the teacher writes something on the board and you just take it from there. But this course requires you to think things out for yourself.
>
> Interviewer: Is that difficult?

Elsie: When I first got here it was, but now I'm learning. I got used to it.

One major difference between learning in the computer lab and learning in the computer science classroom and other similar settings was the extent to which students learn through active trial and error. More than 85% of the students interviewed mentioned this as a distinguishing feature of computer science, and the lab was clearly the setting in which this kind of activity occurred. Students fundamental task in the lab was to figure out how to make their programs perform to accomplish certain goals. Even fairly simple programs often failed to work on the first try. Not surprising, more complicated ones generally required considerable debugging. Once programs worked students were prone to try to improve or elaborate them, thus creating another cycle of improvement through trial and error.

Charlie: I like the satisfaction of doing something that I feel was the best I could do . . . I can make it (the program) look nicer or do something more -- put more extras on it . . . I enjoy that.

This process of trial and error required active engagement on the part of the students which tapped their intellect and imagination in a way they felt other classes generally did not. Student interviews were filled with favorable comments such as the following:

It's different from every other class. (In other classes) you just sit at a desk and you have to do as you are told. Here you do what you are told but you can create things.

The class is fun, but it requires a lot of thinking. I would encourage (others) to take it.

I learn new things everyday. I love it. It's a challenge and I love challenges.

One might expect that working by trial and error would be discouraging to students, especially those who faced error very frequently. However, generally speaking, this did not seem to be the case for a number of reasons. First, it was clear to students that the fact that a program did not work the first time was not some kind of fatal inditement of their skill. Students understood that debugging was a normal part of the creation of a program. Students could observe for themselves that everyone, including the teacher and any "wizards" in their classrooms, often had to struggle to create programs that functioned as they were intended. Second students generally discovered their own errors when they tried out their programs rather than having someone else point them out to them. This, combined with the fact that errors generally did not have negative consequences for students' grades, made errors a signal of a problem to be dealt with rather than an embarrassing failure.

Ned: On the computer if you mess up you can always go back and change a line or fix it. In other classes (like art) once you're done and they grade it if it's wrong it's wrong . . . You might be able to figure out what you did wrong, but it's too late after that. But in computer science you can every so often run the program and see what's wrong and fix it before the teacher grades it. So trial and error is pretty important. It helps you learn what you're doing wrong. It helps you figure out how to fix things up and how to make them . . . right.

Since students were responsible for marshalling the resources necessary to fix their errors, for many students programming was

experienced as a series of personal challenges. This sense of active personal challenge was very motivating to many students.

> Sam: They (students) know they are not doing it (programming) for the teacher really. They are doing it for themselves, seeing the effects of what they put in come back out on the screen and work. They're doing it . . . to try to better themselves. Each program gets better and better.

If students had difficulty in meeting these challenges by themselves, help was readily available, from peers if not always from the teacher, so few students remained stuck on a particular problem for so long that it became really frustrating or created a debilitating sense of failure.

Conclusions

Students' preference for working with computers in the lab rather than learning about them in the classroom was clear. There is no doubt that some of the material covered in the classroom lectures was crucial in helping students achieve their goals in the lab. However, in general, work in that environment was perceived as boring. Students were often inattentive and showed relatively little interest in or enthusiasm for their work in the classroom setting. In contrast, the large majority of the students enjoyed their time in the lab and evidenced much more involvement in their work there. This increase in enjoyment and motivation was due to many factors. As in the GPTutor classes, relations between students and teachers became much more collegial when the students worked on the computers. In sharp contrast to the GPTutor classes, students interacted with each other much more when in the lab than during whole class instruction. Although much of this interaction was purely social, and hence potentially distracting, task-oriented helping interactions were also very common. The freedom to give and

receive help from others served many positive functions, as might be expected from the substantial literature on peer helping. In the lab students were more able to link their work to their own interests and goals than in the classroom which was also very motivating. In addition, the sense of personal challenge created by the active involvement in trial and error learning created an atmosphere conducive to active thinking rather than passive assimilation of knowledge.

References

Becker, H. J. (1986). Instructional uses of school computers. Reports from the 1985 National Survey, Issue 1, Center for Social Organization of Schools, The Johns Hopkins University, Baltimore, MD.

Bogdan, R. C., & Biklen, S. K. (1982). Qualitative research for education: An introduction to theory and methods. New York: Allyn & Bacon.

Buckley, W. M. (1988). Computers failing as teaching aids: Heralded revolution falls short due to lack of machines. Wall Street Journal, p. 17.

Goetz, J. P., & LeCompte, M. D. (1984). Ethnography and qualitative design in education research. Orlando, FL: Academic Press.

Hativa, N., Swisa, S., & Lesgold, A. (1989, March). Competition in traditional CAI: Motivational, sociological and instructional-design issues. In A. Di Sessa (Chair), Computers and classroom social processes. Symposium conducted at the meeting of the American Educational Research Association, San Francisco.

Miles, M. B., & Huberman, A. M. (1984). Qualitative data analysis: A sourcebook of new methods. Beverly Hills, CA: Sage.

Olson, S. (1976). Ideas and data: Process and practice of social research. Homewood, IL: The Dorsey Press.

Quality Education Data (1984, January). Microcomputer data. Unpublished raw data presented to the Naval Materials Council, Dallas, TX.

Schofield, J. W. (1985). The impact of an intelligent computer-based tutor on classroom social processes: An ethnographic study. Unpublished manuscript, University of Pittsburgh, Learning Research and Development Center, Pittsburgh, PA.

Schofield, J. W., Evans-Rhodes, D., & Huber, B. R. (1989). Artificial intelligence in the classroom: The impact of a computer-based tutor on teachers and students (Technical Report No. 3). Pittsburgh: University of Pittsburgh, Learning Research and Development Center.

Schofield, J. W., Evans-Rhodes, D., & Huber, B. R. (1990). Artificial intelligence in the classroom: The impact of a computer-based tutor on teachers and students. Social Science Computer Review, 8, 24-41.

Sheingold, K., Kane, J., & Endreweit, M. (1983). Microcomputer use in schools: Developing a research agenda. Harvard Educational Review, 53, 412-432.

Sheingold, K., Martin, M. W., & Endreweit, M. E. (1987). Preparing urban teachers for the technological future. In R. D. Pea & K. Sheingold (Eds.), Mirrors of minds: Patterns of experience in educational computing (pp. 67-85). Norwood, NJ: Ablex.

Smith, L. M., & Geoffrey, W. (1968). The complexities of an urban classroom. New York: Holt, Rinehart, & Winston.

Strauss, A. (1987). Qualitative analysis for social scientists. Cambridge: Cambridge University Press.

Webb, E. J., Campbell, D. T., Schwartz, R. D., & Sechrest, L. (1966). Unobtrusive measures: Nonreactive research in the social sciences. Chicago: Rand-McNally.

University of Pittsburgh/Schofield

ERIC Facility-Acquisitions
4350 East-West Hwy., Suite 1100
Bethesda, MD 20814-4475

Dr. Debra Evans
Applied Science Associates, Inc.
P. O. Box 1072
Butler, PA 16003

Dr. Beatrice J. Farr
Army Research Institute
PERI-IC
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Elizabeth Fennema
Curriculum and Instruction
University of Wisconsin
225 North Mills Street
Madison, WI 53706

Prof. Donald Fitzgerald
University of New England
Department of Psychology
Armidale, New South Wales 2351
AUSTRALIA

Dr. Michael Flaningam
Code 52
NPRDC
San Diego, CA 92152-6800

Dr. J. D. Fletcher
Institute for Defense Analyses
1801 N. Beauregard St.
Alexandria, VA 22311

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Department of Humanities and
Social Sciences
Harvey Mudd College
Claremont, CA 91711

Dr. Philip Gillis
Army Research Institute
PERI-II
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Mr. Lee Gladwin
305 Davis Avenue
Leesburg, VA 22075

Mr. Harold Goldstein
University of DC
Department Civil Engineering
Bldg. 42, Room 112
4200 Connecticut Avenue, N.W.
Washington, DC 20008

Dr. Sherrie Gott
AFHRL/MOMJ
Brooks AFB, TX 78235-5601

Dr. T. Govindaraj
Georgia Institute of
Technology
School of Industrial
and Systems Engineering
Atlanta, GA 30332-0205

Dr. Dik Gregory
Admiralty Research
Establishment/AXB
Queens Road
Teddington
Middlesex, ENGLAND TW11OLN

Michael Habon
DORNIER GMBH
P.O. Box 1420
D-7990 Friedrichshafen 1
WEST GERMANY

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Mr. H. Hamburger
Department of Computer Science
George Mason University
Fairfax, VA 22030

Dr. Cheryl Hamel
NTSC, Code 711
Orlando, FL 32813

Dr. William R. Murray
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue
Box 580
Santa Clara, CA 95052

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 801
Department of Educational
Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0031

Office of Naval Research,
Code 1142CS
800 N. Quincy Street
Arlington, VA 22217-5000
(6 Copies)

Dr. Judith Orasanu
Basic Research Office
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Nancy N. Perry
Naval Education and Training
Program Support Activity
Code-047
Building 2435
Pensacola, FL 32509-5000

Dept. of Administrative Sciences
Code 54
Naval Postgraduate School
Monterey, CA 93943-5026

Dr. Joseph Psotka
ATTN: PERI-IC
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333-5600

Dr. J. Wesley Regian
AFHRL/IDI
Brooks AFB, TX 78235

Dr. Charles M. Reigeluth
330 Huntington Hall
Syracuse University
Syracuse, NY 13244

Mr. William A. Rizzo
Code 71
Naval Training Systems Center
Orlando, FL 32813

Dr. Linda G. Roberts
Science, Education, and
Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Dr. Janet W. Schofield
816 LRDC Building
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Judith W. Segal
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr. Robert J. Seidel
US Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Randall Shumaker
Naval Research Laboratory
Code 5510
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

Dr. Derek Sleeman
Computing Science Department
The University
Aberdeen AB9 2FX
Scotland
UNITED KINGDOM

Ms. Gail K. Slemon
LOGICON, Inc.
P.O. Box 85158
San Diego, CA 92138-5158

Dr. Alfred F. Smode
Code 7A
Research and Development Dept.
Naval Training Systems Center
Orlando, FL 32813-7100

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Linda B. Sorisio
IBM-Los Angeles Scientific Center
11601 Wilshire Blvd., 4th Floor
Los Angeles, CA 90025

Dr. Marian Stearns
SRI International
333 Ravenswood Ave.
Room B-5124
Menlo Park, CA 94025

Dr. David E. Stone
Computer Teaching Corporation
1713 South Neil Street
Urbana, IL 61820

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1205 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Douglas Towne
Behavioral Technology Labs
University of Southern California
1845 S. Elena Ave.
Redondo Beach, CA 90277

Dr. Frank L. Vicino
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Jerry Vogt
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Thomas A. Warm
Coast Guard Institute
P. O. Substation 18
Oklahoma City, OK 73169

Dr. Beth Warren
BBN Laboratories, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Douglas Wetzel
Code 51
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Barbara White
BBN Laboratories
10 Moulton Street
Cambridge, MA 02238

Dr. David Wilkins
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Marsha R. Williams
Applic. of Advanced Technologies
National Science Foundation
SEE/MDRISE
1800 G Street, N.W., Room 635-A
Washington, DC 20550

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Dr. Merlin C. Wittrock
Graduate School of Education
UCLA
Los Angeles, CA 90024

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Prince of Wales Road
Exeter EX44PT
ENGLAND

Dr. Joseph L. Young
National Science Foundation
Room 320
1800 G Street, N.W.
Washington, DC 20550

Dr. Uri Zernik
General Electric:
Research & Development Center
Artificial Intelligence Program
PO Box 8
Schenectady, NY 12301